Importing Modules

```
# Importing the Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

✓ 3.2s

Loading Dataset

Let's first load the required HR dataset using pandas's read CSV function.

```
df=pd.read_csv('HR_comma_sep.csv')
```

✓ 0.0s

```
df.head()
```

✓ 0.0s Python

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Department	salar
0	0.38	0.53	2	157	3	0	1	0	sales	lo
1	0.80	0.86	5	262	6	0	1	0	sales	mediur
2	0.11	0.88	7	272	4	0	1	0	sales	mediur
3	0.72	0.87	5	223	5	0	1	0	sales	lo
4	0.37	0.52	2	159	3	0	1	0	sales	lo

Let's Jump into Data Insights

In the given dataset, you have two types of employee one who stayed and another who left the company. So, you can divide data into two groups and compare their characteristics. Here, you can find the average of both the groups using groupby() and mean() function.

```
left = df.groupby('left')
left.mean(numeric_only=True)
```

✓ 0.0s

left	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_last_5years
0	0.666810	0.715473	3.786664	199.060203	3.380032	0.175009	0.026251
1	0.440098	0.718113	3.855503	207.419210	3.876505	0.047326	0.005321

Here you can interpret, Employees who left the company had low satisfaction level, low promotion rate, low salary, and worked more compare to who stayed in the company.

The describe() function in pandas is convenient in getting various summary statistics. This function returns the count, mean, standard deviation, minimum and maximum values and the quantiles of the data.

```
df.describe()
```
✓ 0.1s

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	0.021268
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	0.144281
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	0.000000
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	0.000000
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	0.000000
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	1.000000

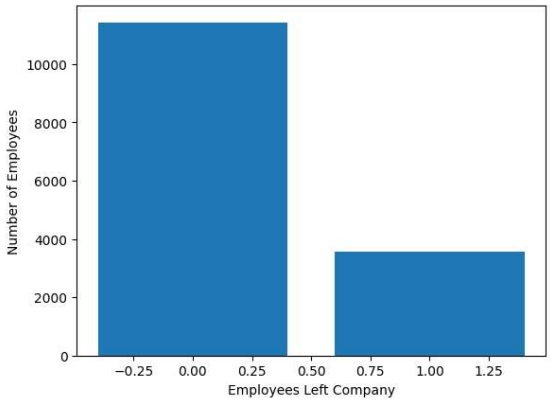
Data Visualization

Employees Left

Let's check how many employees were left?

Here, you can plot a bar graph using Matplotlib. The bar graph is suitable for showing discrete variable counts.

```
1  left_count=df.groupby('left').count()
2  plt.bar(left_count.index.values,left_count['satisfaction_level'])
3  plt.xlabel('Employees Left Company')
4  plt.ylabel('Number of Employees')
5  plt.show()
```
✓ 0.1s



```
df.left.value_counts()
```
✓ 0.0s

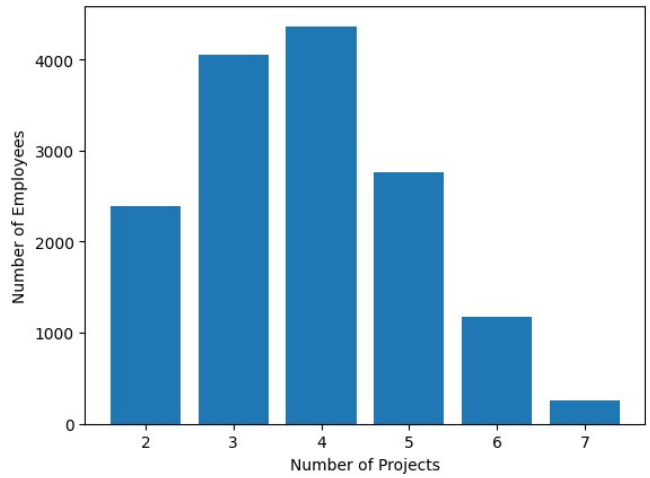
```
left
0    11428
1     3571
Name: count, dtype: int64
```

Here, you can see out of 15,000 approx 3,571 were left, and 11,428 stayed. The no of employee left is 23 % of the total employment.

Number of Projects

Similarly, you can also plot a bar graph to count the number of employees deployed on How many projects?

```
num_projects=df.groupby('number_project').count()
plt.bar(num_projects.index.values,num_projects['satisfaction_level'])
plt.xlabel('Number of Projects')
plt.ylabel('Number of Employees')
plt.show()
```
✓ 0.0s

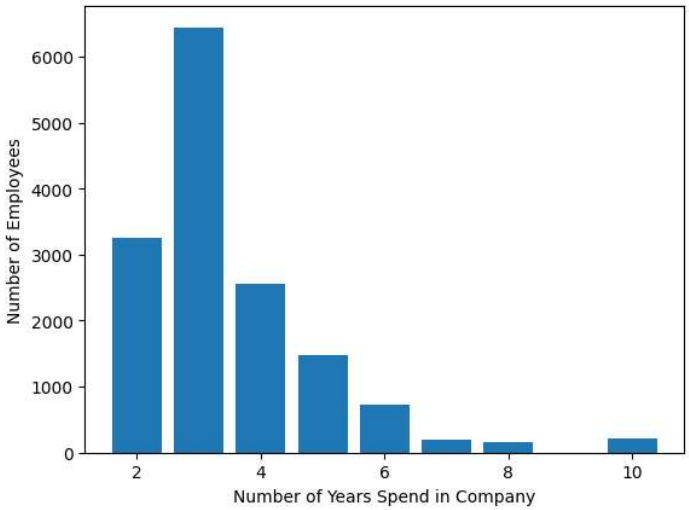


- Most of the employee is doing the project from 3-5.

Time Spent in Company

```
experienced=df.groupby('time_spend_company').count()
plt.bar(experienced.index.values,experienced['satisfaction_level'])
plt.xlabel('Number of Years Spend in Company')
plt.ylabel('Number of Employees')
plt.show()
```

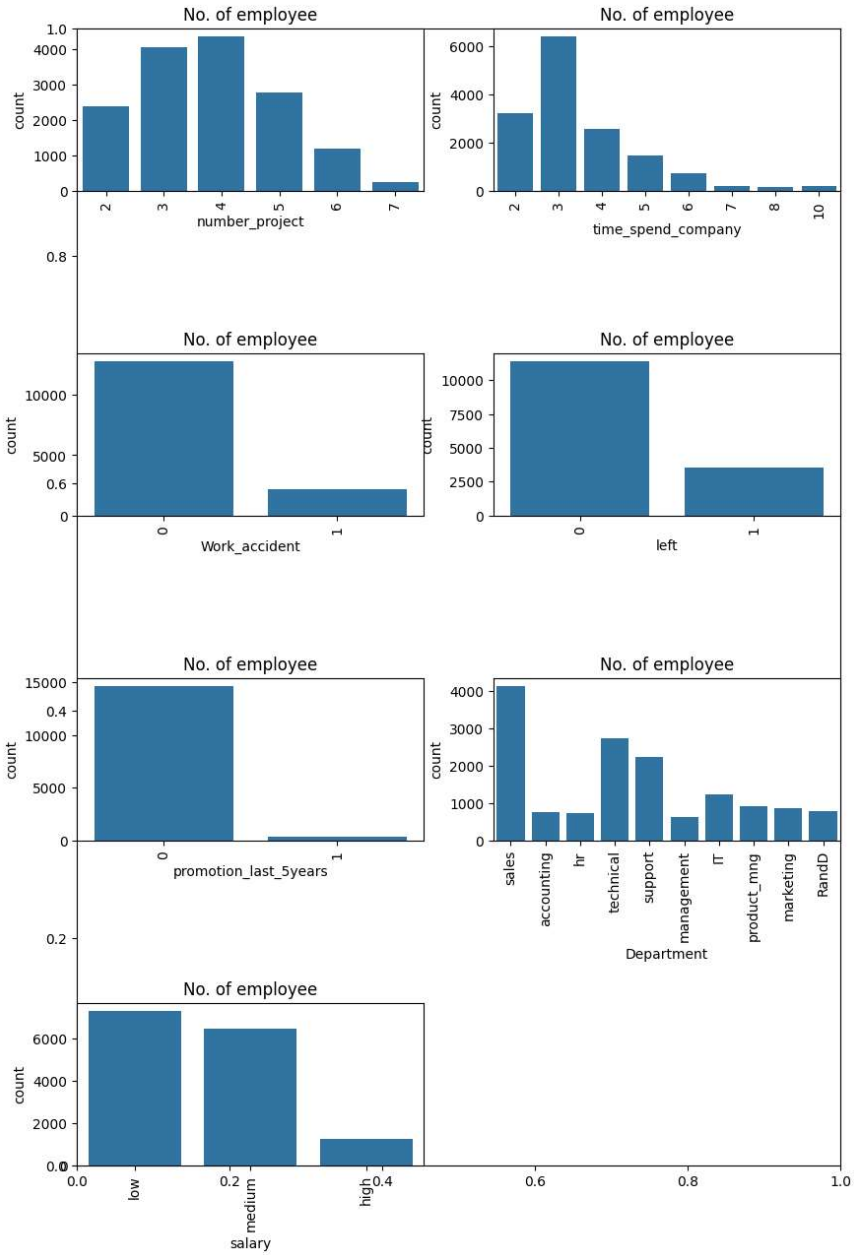
✓ 0.1s



Subplots using Seaborn

```
features=['number_project','time_spend_company','Work_accident','left', 'promotion_last_5years','Department','salary']
fig=plt.subplots(figsize=(10,15))
for i, j in enumerate(features):
    plt.subplot(4, 2, i+1)
    plt.subplots_adjust(hspace = 1.0)
    sns.countplot(x=j,data=df)
    plt.xticks(rotation=90)
    plt.title("No. of employee")
```

✓ 1.2s

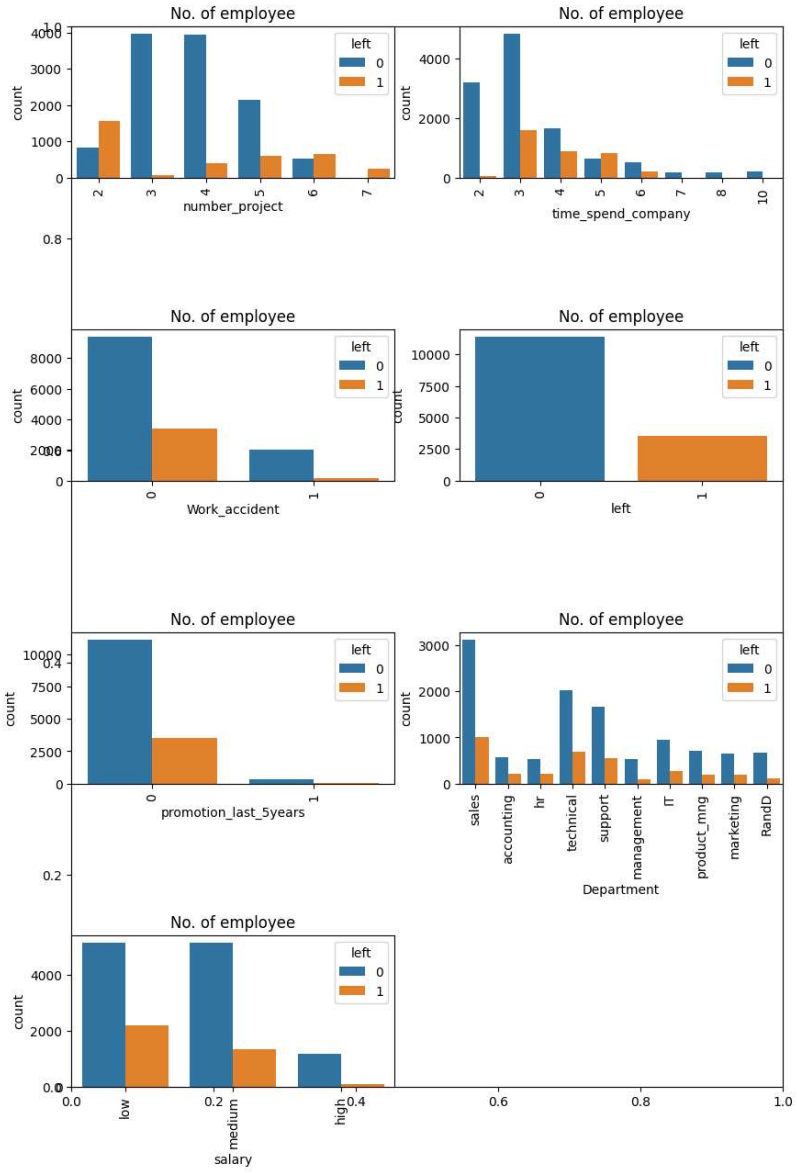


You can observe the following points in the above visualization:

- Most of the employee is doing the project from 3-5.
- There is a huge drop between 3 years and 4 years experienced employee.
- The no of employee left is 23 % of the total employment.
- A decidedly less number of employee get the promotion in the last 5 year.
- The sales department is having maximum no.of employee followed by technical and support
- Most of the employees are getting salary either medium or low.

```
features=['number_project','time_spend_company','Work_accident','left', 'promotion_last_5years','Department','salary']
fig=plt.subplots(figsize=(10,15))
for i, j in enumerate(features):
    plt.subplot(4, 2, i+1)
    plt.subplots_adjust(hspace = 1.0)
    sns.countplot(x=j,data=df,hue='left')
    plt.xticks(rotation=90)
    plt.title("No. of employee")
```

✓ 1.2s



You can observe the following points in the above visualization:

- Those employees who have the number of projects more than 5 were left the company.
- The employee who had done 6 and 7 projects, left the company it seems to like that they were overloaded with work.
- The employee with five-year experience is leaving more because of no promotions in last 5 years and more than 6 years experience are not leaving because of affection with the company.
- Those who promotion in last 5 years they didn't leave, i.e., all those left they didn't get the promotion in the previous 5 years.

- **Data Analysis and Visualization Summary:**

Following features are most influencing a person to leave the company:

- Promotions: Employees are far more likely to quit their job if they haven't received a promotion in the last 5 years.
- Time with Company: Here, The three-year mark looks like a time to be a crucial point in an employee's career. Most of them quit their job around the three-year mark. Another important point is 6-years point, where the employee is very unlikely to leave.
- Number Of Projects: Employee engagement is another critical factor to influence the employee to leave the company. Employees with 3-5 projects are less likely to leave the company. The employee with less and more number of projects are likely to leave.
- Salary: Most of the employees that quit among the mid or low salary groups.

- **Building a Prediction Model**

Pre-Processing Data

```
catcol=df.select_dtypes('object').columns
```
✓ 0.0s

```
catcol
```
✓ 0.0s

```
Index(['Department', 'salary'], dtype='object')
```

```
from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder()
df[catcol]=oe.fit_transform(df[catcol])
```
[15] ✓ 0.2s

```
df.head()
```

0.0s

Python

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Department	salary
0	0.38	0.53	2	157	3	0	1	0	7.0	1.0
1	0.80	0.86	5	262	6	0	1	0	7.0	2.0
2	0.11	0.88	7	272	4	0	1	0	7.0	2.0
3	0.72	0.87	5	223	5	0	1	0	7.0	1.0
4	0.37	0.52	2	159	3	0	1	0	7.0	1.0

+ Code + Markdown

Split Train and Test Set

```
#Spliting data into Feature and
X=df[['satisfaction_level', 'last_evaluation', 'number_project',
      'average_montly_hours', 'time_spend_company', 'Work_accident',
      'promotion_last_5years', 'Department', 'salary']]
y=df['left']
```

0.0s

+ Code + Markdown

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(xtrain,ytrain)
ypred=lr.predict(xtest)
```

0.0s

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
ac=accuracy_score(ytest,ypred)
cm=confusion_matrix(ytest,ypred)
cr=classification_report(ytest,ypred)
print(f'Accuracy:{ac}\nConfusion Metrix:\n{cm}\nClassification Report:\n{cr}')
```

0.0s

```
Accuracy:1.0
Confusion Metrix:
[[3416    0]
 [   0 1084]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3416
           1       1.00      1.00      1.00      1084

    accuracy                           1.00      4500
   macro avg       1.00      1.00      1.00      4500
weighted avg       1.00      1.00      1.00      4500
```

```
train=lr.score(xtrain,ytrain)
test=lr.score(xtest,ytest)
print(f"Trainning Score:{train}\nTesting Score:{test}")
```

[53] ✓ 0.0s

```
Trainning Score:1.0
Testing Score:1.0
```